A Real-Time Community-of-Interest Framework for Command-and-Control Applications

Ray Paul
Department of Defense
Washington, DC
raymond.paul@osd.mil

W. T. Tsai
Department of Computer Science and Engineering
Arizona State University
Tempe, AZ 85287
wtsai@asu.edu

Abstract

The paper presents a real-time Community of Interest (COI) framework for Command and Control (C2) applications. The overall goal is to support superior decision making, COIs' flexible synchronization and collaboration, and real-time communication. The extended service-oriented architecture (SOA) based framework consists of three layers including mission layer, COI service layer and support layer. Mission service layer consists of a set of mission/tactical services. COI service layer provides COI coordinator service, policy specification/evaluation services, situation awareness and dynamic reconfiguration services. Overall scenarios of COI's services are described to illustrate the relationships among the services. The support service layer consists of discovery services, messaging services, security service, scenario specification services, quality-of-assurance service, data storage/retrieval services, and maintenance services. This paper focuses on COI service layer. A prototype is developed to demonstrate the main proposed techniques, approaches and the process using an example COI of bookstores and publishers. It has three layers including data collection/filter layer, analysis layer and presentation layer.

Keywords: Community of interest, COI services, Coordinator services, Mediator services, Collaboration services, Situation awareness, Data classification, Service mining

1. Introduction

Community of Interest (COI) is receiving significant attention recently as it is used as a principal component for future C2 systems such as GIG (Global Information Grid) and NCES (Net Centric Enterprise Systems). A COI is a place where interested parties can get related information in real time, and each party can contribute to the discussion and share information with each other. In this way, a DoD commander can receive the needed information, download the related knowledge, participate in decision making, make

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **JUN 2004**	2. REPORT TYPE	3. DATES COVERED **00-00-2004 to 00-00-2004**

4. TITLE AND SUBTITLE **A Real-Time Community-of-Interest Framework for Command-and-Control Applications**	5a. CONTRACT NUMBER 5b. GRANT NUMBER 5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER 5e. TASK NUMBER 5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Arizona State University,Department of Computer Science and Engineering,Tempe,AZ,85287**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S) 11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**18**	

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std Z39-18

strategic, operational, and tactical plans, issue commands to soldiers, and monitor progress in real time [2].

This paper will propose a COI framework for C2 applications in a network-centric environment. This framework has the following features:

- **Service-Oriented Architecture (SOA):** As DoD is embracing SOA for its applications such as NCES and GIG Enterprise Services (GES), the COI will be implemented using SOA because an application running on SOA is inherently survivable.
- **Situation awareness and analysis**: Due to the specialized nature of COI, each COI has its own situation assessment algorithms to determine the current situation. The COIs also allow participants to contribute and present new situation assessment algorithms to meet the changing environment.
- **Real-time messaging and alerts:** The proposed COI framework is a real-time application where commanders participating in COIs can send and receive time-critical messages, and each COI can send and receive real-time alerts to and from other COIs.
- **Fault-tolerant computing**: The COI needs to be survivable in case of enemy's attacks and must be able to recover occasional failures by employing dynamic reconfiguration;
- **Database support:** Each COI has its own database, can receive, process, filter, and retrieve information from its database in real time, and it can send its data to other concerned COIs.
- **Knowledge engineering and processing:** The data stored in the COI databases must be processed for easy retrieval and application. A processed knowledge is knowledge that be used, interpreted and applied.
- **Data mining:** A specific technique for knowledge processing is data mining where abnormal data, trends, statistical characteristics, relationships among data can be automatically classified and detected by powerful searching and analysis tools for C2 decision-making;
- **Network computing**: Each COI will run on a network-centric enterprise system for interoperability, survivability and knowledge sharing;
- **Multimedia:** Each COI will have a multimedia presentation with video, photos, voices, map, pictures and texts. These data need to be transmitted reliably and securely over a DoD enterprise network.

The paper is organized as follows. Section 2 illustrates the COI architecture and describes the main COI services. Section 3 demonstrates a prototype. Finally, Section 4 concludes the paper.

2. COI Architecture

Each COI consists of core services that represent the main capabilities and common services that are shared among most COIs (Figure 1). The architecture is divided into three layers with the services at a higher layer using services of the lower layers.

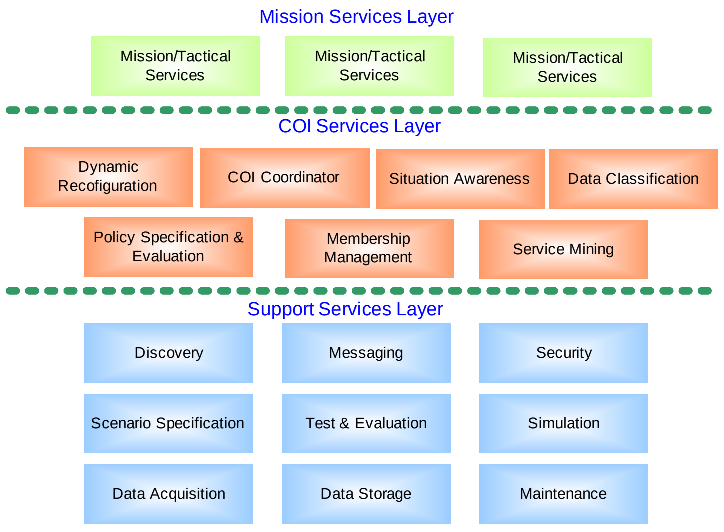

Figure 1. COI Architecture

The Mission Services Layer contains mission-specific services for specific C2 applications. Modern warfighting is agile, and these services need to change at runtime in real time when new services are created and obsolete service become deactivated. Services in this layer provide individualized COI services for warfighting and they exchange data, information, decisions, knowledge and alerts with each other to achieve the overall mission.

The COI Service Layer contains common services needed to implement specific C2 COIs. These common services include membership management, C2 policy management, dynamic reconfiguration, and situation awareness management.

- Membership Management Service (MMS): The service provides mechanisms for users to register/un-register with specific COIs, and also manages connection among individual COIs. A COI may register with another COI to get real-time data and alerts, a COI may be a supervisor of another COI, and a steady stream of data will flow from one COI to another COI. In fact, a COI may have other relationships with other COIs, and this is discussed in Section 2.2.

- Policy Specification/Evaluation Service (PSS) [13]: The proposed COI framework supports policy-based computing where C2 policies are specified and used in various computing including situation awareness and dynamic reconfiguration. These policies will be specified using a Policy Specification and Execution Language (PSEL), and policies stated in this language can be executed in real time at runtime to determine system properties such as security and reliability. PSES allows decision makers and commanders to issue appropriate commands without specifying the details of the command. C2 policies in this framework can be changed at runtime by participating COIs, and because computing is based on policies, thus once policies are changed, the behaviors of the system are changed (See [6] for details).

- Situation Awareness Service (SAS): This kind of services collect data from sensors and other participating services, analyze the data received, and issue alerts to concerned COIs and participants.
- Dynamic Reconfiguration Service (DRS): The service supports dynamic reconfiguration in a SOA in real time at runtime [6]. The existing SOA such as Microsoft's .NET allows runtime service location, binding and execution. However, once a decision is made, the same service will be called over and over again unless another round of service discovery and binding is performed, and this must also be initiated by an attendant. The proposed COI framework is different as it has DRS that can continuously monitor each participating service, and initiate a reconfiguration process, and implement the process using the stated C2 policies. With DRS, individual service can be added, removed, and replaced at runtime without interrupting the system operations.
- Service Mining Service (SMS): During reconfiguration, new services must be identified to replace the failed service. This can be done by mining the existing services to find an effective mapping between a request and the existing services.
- Data Classification Service (DCS): This use the data mining approach to classify the data collected into appropriate categories for decision making [3][4].

The Support Service Layer provides basic services needed to support common COI services. Many of these services provide similar functionalities as those of core enterprise services in NCES. As the proposed COI framework has the dynamic reconfiguration capabilities, many of these basic services need to have real-time behavior and must be involved in continuous monitoring and data collection.

- Discovery Service (DIS): The service provides discovery in traditional SOA such as .NET. However, in addition to collecting static information such as functionality and interface of services in an SOA, these discovery services also need dynamic information about available services in the network including their operational profile, current status (not-activated, idle and ready, working, busy, overloaded, and failed), location, security (unclassified, secret or top secret), scheduled tasks, maintenance schedule, and backup data information.
- Messaging Service (MES): The service provides event-driven messaging to COIs triggered by services such as policy, discovery, situation awareness & dynamic reconfiguration, COI coordinators, security, and storage. The events form complex hierarchy with attributes of reliability, survivability, security & non-repudiation, and timely delivery.

- Security Service (SES): The service ensures system security including encryption, authentication, filtering, and security policy enforcement.
- Scenario Specification Service (SSS): The service allows users to specify system requirements in a scenario specification language and ACDATE model [8][9]. The scenario/ACDATE model is useful for system verification including runtime verification and monitoring, simulation, C2 policy enforcement and static analysis such as completeness and consistency checking.
- Test and Evaluation Service (TES): Scenario-based test supports automated test case/scripts generation using pattern-driven techniques and state models as well. It also supports distributed test execution.
- Simulation Service (SIS): This kind of services simulate system behaviors once the system scenarios are specified using the ACDATE model, and allows various analyses such as security analysis, completeness and consistency analysis, concurrency analysis, timing analysis to be performed at runtime in real time.
- Data Acquisition Service (DAS): The service collects data for situation awareness analysis, runtime monitoring and evaluation, security analysis, and test & evaluation.
- Storage Service (STS): The service provides data storage and retrieval capabilities.

2.1 COI Action Initiations

COI services may be initiated in following ways:

1) Mission-driven activities: This may happen when a commander issues a new C2 policy in a COI, and such change of the C2 policy will immediately trigger a series of actions in the COI framework. For example, when a commander decides that it is necessary to react to a specific situation detected earlier, the commander can update the relevant C2 policies, and the updated C2 policies will alert other commanders in other COIs to react accordingly.
2) Situation-driven activities: This kind of activities starts with a detection of a concerned situation such as abnormal/suspicious/attack behaviors based on data received from data collection services or sensors. These data must be analyzed by the situation awareness services or COIs according the stated C2 policies. Once such behaviors are detected, the system will respond to these behaviors according the previously stated C2 policies.
3) Collaboration-driven activities: This kind of activities will be generated based on collaboration among commanders and decision makers participating in various COIs. Note that multiple commanders may participate in one COI, and multiple

COIs may collaborate with each other with different collaboration strategies (such as peer-to-peer or supervisor-to-subordinate) under different C2 policies.

2.2 COI Coordinator Services

The COI coordinator is a specific service that provides COI coordination including mediation within a COI, collaboration among COIs, and overall optimization in terms of benefit/cost analysis under C2 policies and constraints. Each COI has at least one COI coordinator and this is shown in the following figure.

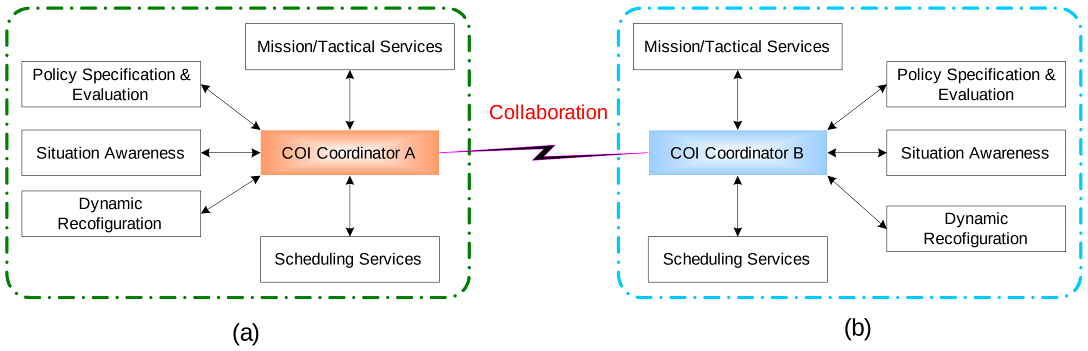

Figure 2. Overall COI Scenarios

Mediator Services within a COI

The COI coordinator needs to interact with other services such as PSS, SAS and DRS, within the COI. The mediation can follow the blackboard model or priority model. In the blackboard, participating services will interact with each other by sharing a common working space, and each service will provide its service upon receiving specific data in the working space [7]. The priority model assigns priorities to each service, and participating services will get its turn for execution according to the priorities [1]. The priorities may be changed at runtime by C2 policies so that the system will have different behaviors at different times, e.g., peacetime or wartime. The COI coordinator may also work with a hybrid model, i.e., integrating both blackboard and priority model. This kind of mediation promotes loose coupling by keeping the services from referring to each other explicitly regarding COI activities.

Collaboration Services

The proposed COI framework is a distributed network with individual COIs on a secure DoD network. Each COI focuses on a specific area of applications or data, and each COI has its own unique data and knowledge processing rules, deductive capabilities, filtering, and classification rules for its specific data. However, these COI also can cooperate with each other by providing real-time alerts and data to fellow COIs in the network as shown in Figure 3 below. COIs may need to register with other COIs to get their real-time alerts and updates.

The relationship among COIs can be different, for example:

- Supervisor-to-subordinate relationship: The supervisor COI can send command to and impose control policy on the subordinate COI, which needs to report the results back. The relationship is *asymmetric* and *transitive*. That is if COI_1 is a supervisor of COI_3, then COI_3 only can be a subordinate of COI_1. The relationship cannot be reversed. Assume that COI_3 is the supervisor of COI_{n-1}, then COI_1 is also a supervisor of COI_{n-1}. This relationship is transitive.
- Peer-to-Peer: Each COI has unique characteristics and provides specific services to the other. They may cooperate with each other to achieve common goals, than compete for resources and tasks. The relationship is *symmetric* and *transitive*.
- Competitor-to-competitor: Each COI has similar characteristics and provides similar capabilities regarding the requests. However, they may have their own goals, and compete for the limited resources. The relationship is *symmetric* and *transitive*.
- Winner-to-Winner: Although each COI may achieve its goal by itself, the cooperation with other COIs provides better solutions. The goals are independent, and without resource competition. Mathematical game theory can be utilized to maximize each player's goal or minimize each player's loss. The relationship is *symmetric* and *transitive*.

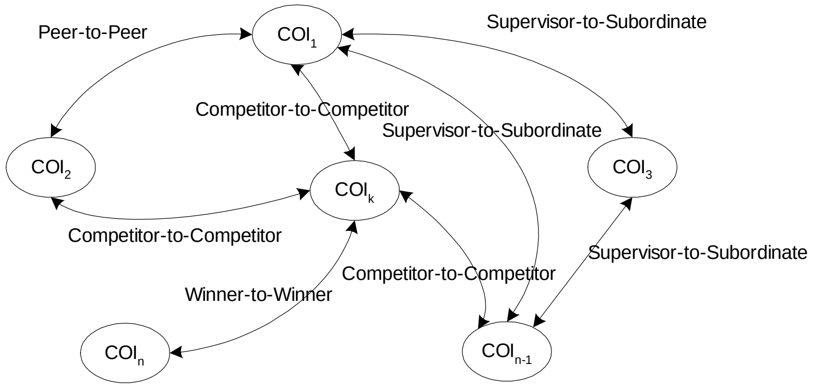

Figure 3. Net-centric COIs

2.3 Data Classification

The first step of classification is to build the model to tune the parameters in the classification rules using training data with the classification algorithms (Figure 4). In the second step, the accuracy of the rule-based model needs to be estimated. The *holdout method* is a simple technique that uses a test set of class labeled samples. These samples are randomly selected and are independent of the training samples. The accuracy of a model on a given test set is the percentage of test set samples that are correctly classified by the model. If the accuracy of the model is considered acceptable, the model can be used to classify future data for which the class label is not known.

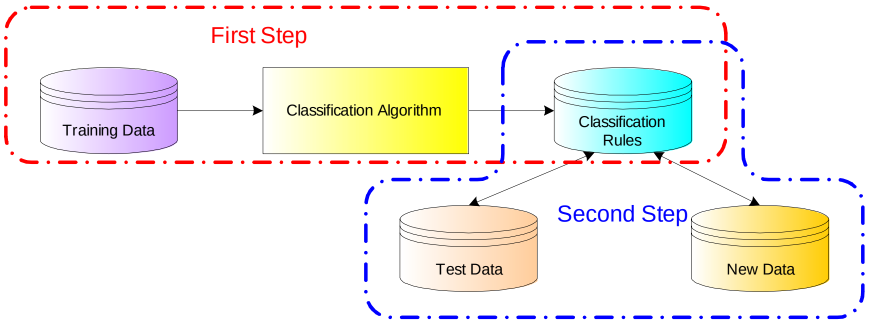

Figure 4. Data classification process

Data can be classified according to the attributes, such as frequency of sending/receiving (Sparse, Often, Intensive), level (Low, Guarded, Elevated, High, Severe), source (Top, Peer, Subordinate), and type (Warning, Alerting, Mandatory). The model is constructed by analyzing data stored in database. The data tuples collectively form the training data set used for a better understanding of the data contents. In the second step, the model is used for classification.

A typical warfighting scenario with classification services using data mining techniques is shown as follows:

1. The sensor probes suspicious signals from certain location, and posts the data to distributed database using database storage services, and to profiles maintained by discovery services.
2. Discovery services match events/signals (producer) to listeners (consumer) with the profiles, prepare the trigger alert messages and send to messaging services.
3. The messaging services notify the consumer with the messages in real-time.
4. The notified edge user pulls the data from the distributed database. The storage services provide the delivery with integrity and in timely fashion.
5. Before taking any action, the pulled data need to be analyzed and interpreted.

2.4 Service Mining

During reconfiguration, new services must be identified to replace the failed service. This can be done by mining the existing services. Service mining is different from the traditional data mining [5] in several ways: 1) It deals with runtime services rather than static data, because each service needs not only static (e.g., functionality) but also dynamic information (e.g., capability, binding, and interoperability), this complicates the mining process; 2) the selected services may need to be verified at runtime before they can be used to ensure that the service keeps the same level of performance; 3) If the failed service has some state information, it is necessary for the replacement service to start from this particular state before resuming the computation. In other words, the distributed monitoring agent is also responsible to keep track of the state information for the monitored service, and in case of reconfiguration, the agent retrieves the saved state for the replacement service.

The **service-mining** problem can be described as finding an effective mapping between a request and the existing services. Given the large number of services available in a network, it is necessary to organize the existing services into a service hierarchy (or a service tree) to have an efficient mapping between a request and services. The following are key issues in service mining:

- Creating a service tree by applying hierarchical clustering algorithms to services;
- Extending the service tree with new services by designing incremental learning algorithms for hierarchical clustering; and
- Refining the service tree by reorganizing services to optimize the hierarchy with a compact structure.

2.5 Situation-Awareness Service (SAS)

An SAS consists of a SA decision manager (SADM), which specifies reconfigurable policy/rules for situation analysis and communicates with other services (e.g., COI Coordinator); and a set of distributed SA action managers (SAAM), each of which controls a group of sensors or any data collectors by gathering and fusing data input. Figure 5 shows the Situation Awareness services architecture.

- SADM receives and analyzes high-level command/control messages or COI-related requests, and specifies the types or categories of situations/data it needs to collect and to support decision making; and determines the process/inference/prediction rules for situation analysis, which demands domain-specific knowledge. SAAM performs a variety of analyses on the rules including completeness and consistency [12], and priority assignment. The rules are stored into database in such a way that they are easy to be retrieved.

- The distributed SAAMs extract the reconfigurable rules and create detailed SA actions executed on distributed sensors/monitors, including tracking targets, frequency of scanning/tracking data, which are posted and made available in GIG. Also SAAMs need to handle the security of sensor data, including acquisition, calculation, and transmission.

- Once SADM fuses and aggregates information from distributed SAAMs, it may need to use other services, such as Scheduling services for scheduling to handle detected events, Simulation for execution/evaluation of rules, Verification services for rules verification and validation, and Data Mining services for prediction.

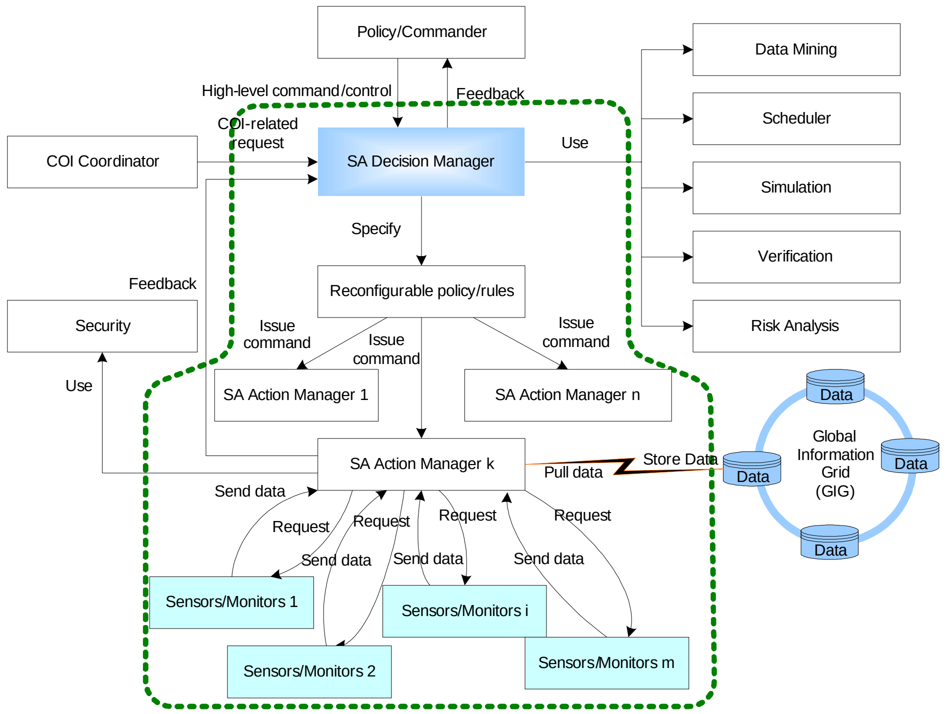

Figure 5. Situation Awareness Services Architecture for Net-centric COIs

Table 1 and Table 2 in the Appendix summarize the capabilities, functionalities and technologies used in COI service layer and support layer, respectively.

3. Prototype of COI

A prototype model for COI has been developed. The model enables autonomous as well as collaborative decision making. Each COI focuses on a specific area of applications or data, and each COI has its own unique data and knowledge processing rules, deductive capabilities, filtering, and classification rules for its specific data. So each COI is decentralized for reliability and survivability, and hence autonomous. COIs cooperate with each other by providing real-time alerts and data to fellow COIs in the network. Participants of COIs can make coordinated C2 decisions to achieve a mission. Also, if needed, a COI can delegate tasks and responsibilities to other COIs and take a decision depending on the results from the COIs.

3.1 Extended SOA for COI model

The current WS architecture does not address real time aspects and dynamic reconfiguration mechanism. Specifically, the DoD enterprise C2 system must address reliability, safety, security, and performance aspects that commercial WS architecture does not address at this time. Also, the DoD enterprise C2 systems must address the system as well as the network aspects in addition to just the application aspects as in the commercial WS architecture. These additional requirements make the proposed SOA unique for DoD C2 applications. In the proposed SOA model [6] each service will be specified using the Interface, Scenario, Constraints (ISC) convention defined as follows:

- **Interface Specification**
  - Input/Output parameters
  - Communication protocols
  - Interfaces with other sub-systems
- **Scenarios Specification**
  - Specify the service scenarios using the ACDATE model
  - Specify interactions with other sub-systems
- **Constraints Specification**
  - Based on the scenario model and ACDATE model
  - Specify the properties of the services must have such as
    - Reliability, availability, security, timing, concurrency, performance, sequence, safety
    - These constraints must be addressed at runtime to assure dependable computing.

The COI model is implemented using SOA so that all of its internal layers are organized into the service architecture. In the SOA, every computing unit is a service and each service is treated the same. All the services in the COI group publish their interface, adhere to constraints and also need to specify the operational scenarios of each service. Each service in the COI group will be specified using the ISC convention.

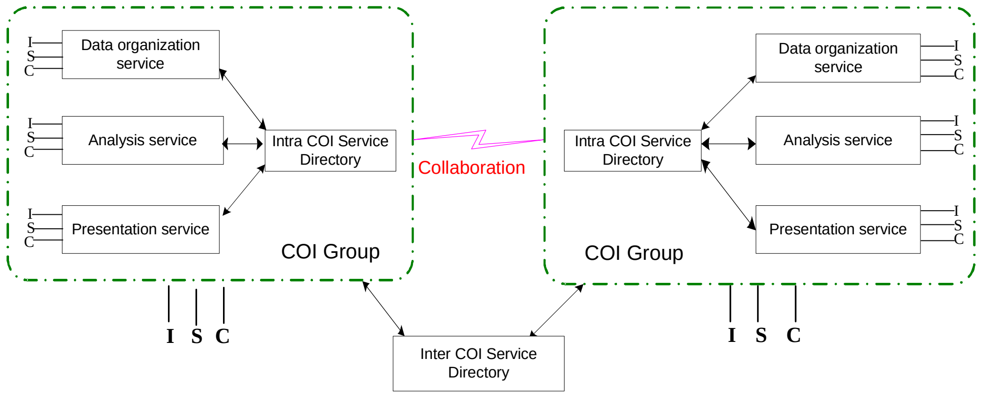
Data organization service
Analysis service
Presentation service
Intra COI Service Directory
COI Group
Collaboration
Intra COI Service Directory
Data organization service
Analysis service
Presentation service
COI Group
I S C
Inter COI Service Directory
I S C

Figure 6. Extended SOA with ISC for COI

Note that a service can be formed by composing several other services, and in this case, the overall service has its own ISC specifications, and each of its sub-services also has its own ISC specifications. In our model, the COI group takes the services of data organization, analysis and presentation layers using the service directory. The service directory publishes all the services that are available in the COI group. Each COI group is itself a service, implemented as an autonomous unit, and can utilize the services of other COI groups if necessary. The COI group A can take the services of other service providers from a different COI group B in case of disruption or failure.

3.2 COI Prototype Design

The prototype developed has three layers: Data Organization Layer, Analysis Layer, and Presentation Layer as shown in the following figure.

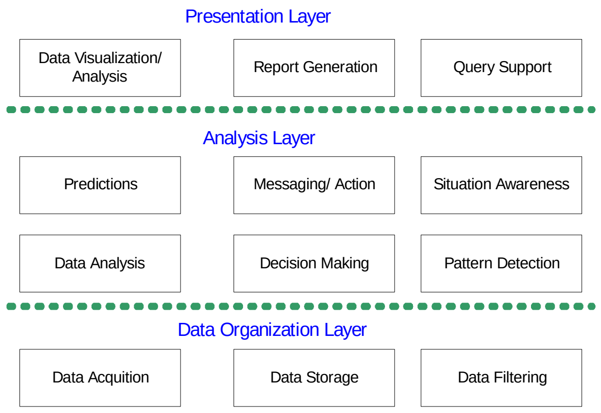

Figure 7. Functional View of Prototype

The following describes the three layers.

Data organization layer: This layer deals with data collection, filtering and storage. Data are collected from different sources and filtered based on filtering rules. Domain data set is passed to the analysis layer. This layer uses open source software tool called "Weka" [16] for data filtering and data classification.

Analysis Layer: This layer has three main functional blocks.
Classification block: This block is used to classify the data. Popular techniques used in data mining for classification are decision trees and clustering. Classification rules are obtained after classifying data. First, training data is fed to the classification block to train it in the desired classification. The block should classify data based on the training. After classification is complete, the information is passed on to the decision module.

Decision module: The inputs to this module are data from the classification block, historical information, situation awareness information, and information from other COIs. This module finds useful patterns in the data and stores them in the pattern repository. It lists all the patterns identified in the data. Also, it generates the list of possible actions that the COI need to take, and messages/alerts to other COIs.

Pattern repository: This block stores patterns. Basic patterns are identified and stored in this repository in advance. The decision module, after identifying the patterns in the data, compares them with the patterns in the repository and, if not present, stores them in the repository.

Presentation Layer: This layer provides a user interface to display the results of the data analysis. The following functionalities are available.
Data visualization and analysis: The data and interesting dependencies in data are represented in the form of tables, charts.
Report generation: The possible actions to be taken by COI, messages and alerts to other COIs are represented visually.
Query support: User query results are presented here.

3.3 Data Filtering

Data collected are often incomplete, noisy, and inconsistent. Data needs to be preprocessed so as to improve the efficiency and ease of mining process. Some of the data preprocessing techniques are data filtering (data cleaning), data integration, data transformations, data reduction. Data filtering routines attempt to fill missing values, smooth out noise, and correct inconsistencies in the data. Some of the basic methods employed for data filtering are filtering data based on missing values, noisy data and inconsistent data. In the data organization layer of the COI, Weka tool is used for data filtering.

Domain Filters: Domain filters are defined to filter data based on the scope and functionality of COI. Every COI has its own domain policy specified. Filtered data from the Weka tool can be run through domain filters to get the domain specific data.

3.4 Example

A hypothetical bookstore is used as an example to implement the proposed COI model. It has customers, bookstore and book publishers, and is implemented on .NET platform [14]. Services of COI groups for tracking customer behavior, next purchase, purchase interval, tracking old and new books and tracking bookstores are implemented as shown in the following figure.

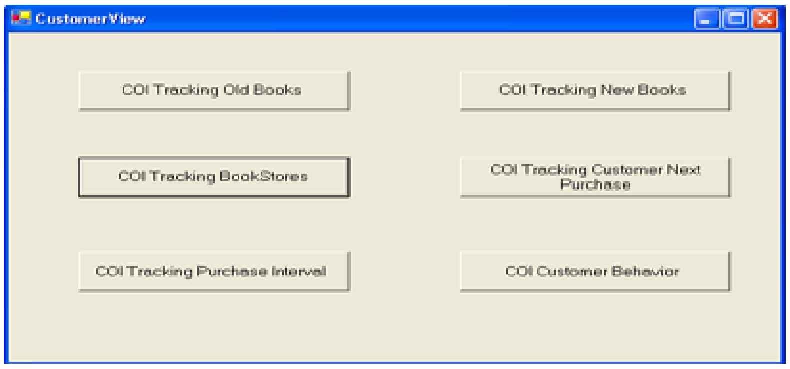

Figure 8. COI Groups in the Bookstore Example

After data are collected, they are filtered using the Weka tool, which is a collection of machine learning algorithms for data mining. Decision tree algorithm is used for classification. The following figure illustrates the filtering process using *Remove Missing Value* filter in Weka.

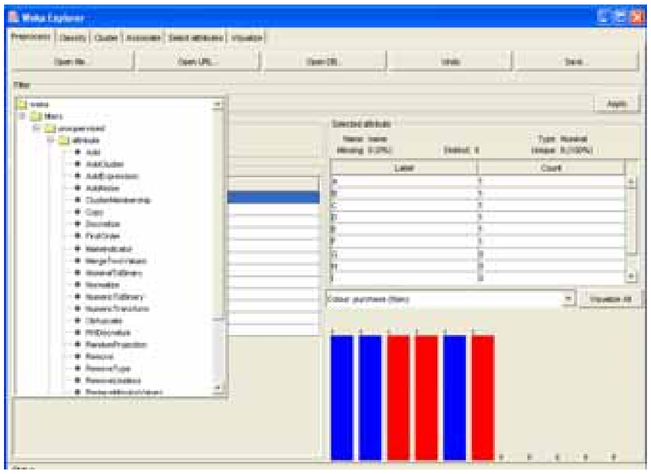

Figure 9. Data Filtering using Weka

A brief description of the some of the scenarios is given here. A scenario where customer needs to buy a new book is considered. Customer objective is to find the book for lowest price. Customer requests the COI group "Tracking new books". This COI uses the "Data organization service" which collects the data about the book requested by the customer from different bookstores. The "Analysis service" finds the best bookstore for the customer to buy the book by taking the book prices/discounts, historical data, and information/alerts from other COIs into consideration. The results are presented to the user using the "Presentation service". The following figure shows the results of presentation service giving the comparison of book prices in different stores.

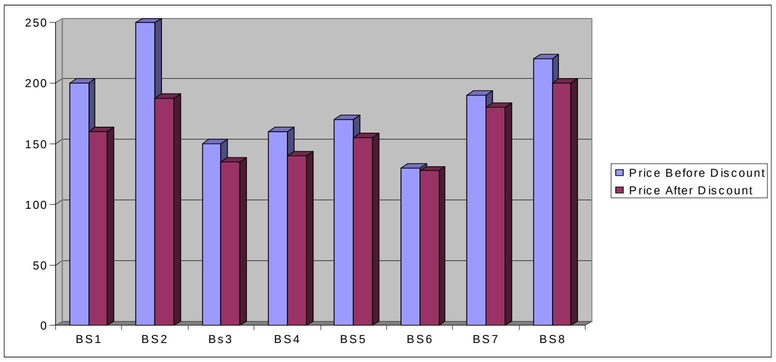

Figure 10. Comparison of Book Prices

A scenario where bookstore needs to track customer behavior in various age groups is considered. Bookstore contacts the "Customer behavior" COI. This COI communicates with "Tracking customers next purchase" and "Tracking customer purchase interval" COIs to make a coordinated decision and returns the analysis results to the bookstore. Domain filters are used to get the data required to a particular age group. This scenario illustrates the coordinated decision making using multiple COI groups. Also, scenarios where publisher needs to track book sales in the bookstores, and customers requesting old and new book information are implemented. The following figure shows the implementation of COI services and the class view in .NET.

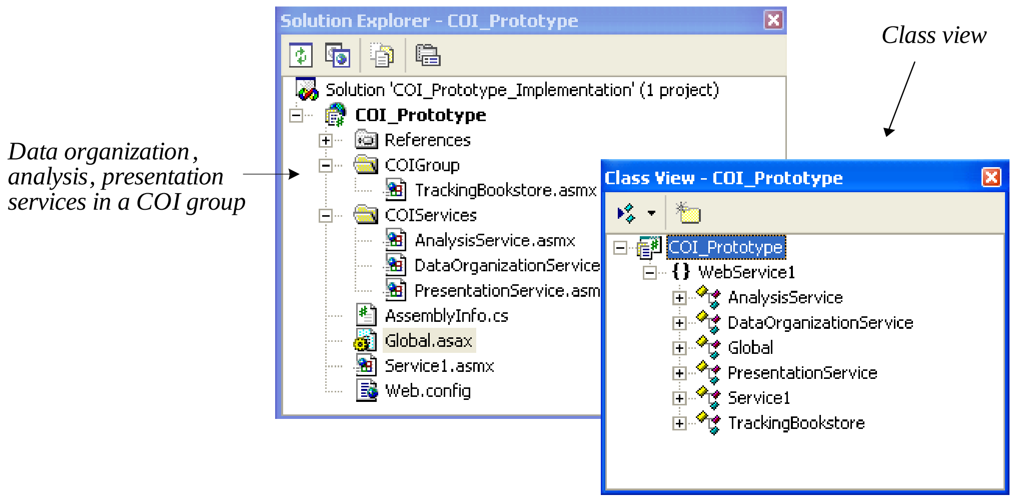

Figure 11. Services and Class View

4. Conclusion and Future Research

A SOA based real-time COI framework for C2 applications, which consists of three layers including mission layer, COI service layer and support layer, is described. This paper focuses on COI service layer. A prototype is developed to demonstrate the main techniques using COIs of bookstores and publishers. Future work includes exploring the algorithms to address collaboration, synchronization, real-time scheduling, security and privacy protection, scalability, survivability, and service mining.

Reference:

[1] A.D. Baker. A survey of Factory Control Algorithms That Can Be Implemented in a Multi-Agent Heterarchy: Dispatching, Scheduling, and Pull. Journal of Manufacturing Systems, Vol. 17, No. 4, 1998, pp. 297-321.

[2] L. Diedrichsen, "Command and Control: Operational Requirements and System Implementation", Information and Security, Volume 5, 2000.

[3] S. Krishnaswamy, A. Zaslavsky, S. W. Loke, "An Architecture to Support Data Mining Services in E-Commerce Environments", Second International Workshop

on Advance Issues of E-Commerce and Web-Based Information Systems (WECWIS 2000), June 2000.

[4] S. Krishnaswamy, A. Zaslavsky, and W. S. Loke, Towards Data Mining Services on the Internet with a Multiple Service Provider Model: An XML Based Approach, Electronic Commerce Research (Special issue on Electronic Commerce and Service Operations), 2(3), August 2001.

[5] H. Liu, A. Mandvikar, P. Foschi, and K. Torrkola. “Active Learning Using Ensembles for Image Mining”. In Proceedings of IJCAI, Acapulco, Mexico, 2003, AAAI Press.

[6] R. P. Paul, W. T. Tsai, “Service-Oriented Architecture for Command and Control Systems with Dynamic Reconfiguration”, to appear in Proc. of 2004 Command and Control Research and Technology Symposium.

[7] N. M. Sadeh, “A Blackboard Architecture for Integrating Process Planning and Production Scheduling”, Concurrent Engineering: Research and Applications, vol. 6, No.2. pp. 88-100, 1998.

[8] W. T. Tsai, L. Yu, F. Zhu, R. Paul, "Rapid Verification of Embedded Systems Using Patterns", Proc. of IEEE COMPSAC, 2003, pp. 466-471.

[9] W. T. Tsai, L. Yu, R. Paul, C. Fan, X. Liu, Z. Cao, “Rapid Scenario-Based Simulation and Model Checking for Embedded Systems”, Proc. of 7th IASTED International Conference on Software Engineering and Applications, 2003, (SEA2003), pp. 568-573.

[10] W. T. Tsai, A. Saimi, L. Yu, R. Paul, “Scenario-based Object-Oriented Test Frameworks”, Proc. of 2003 Third International Conference on Quality Software (QSIC03), pp. 410-417.

[11] W. T. Tsai, C. Fan, R. Paul, and L. Yu, “Automated Event Tree Analysis from Scenario Specifications”, Proc. of IEEE ISSRE, 2003, pp.240-241.

[12] W. T. Tsai, R. Paul, L. Yu, X. Wei, and F. Zhu, “Rapid Pattern-Oriented Scenario-Based Testing for Embedded Systems” to appear in *Software Evolution with UML and XML*, edited by H. Yang, 2004.

[13] W. T. Tsai, “Scenario-based Policy Specification and Execution Language”, Technical Report, Arizona State University, 2004.

[14] A. Troelsen, *C# and the .NET Platform*, Addison Wesley, Reading, MA 2001.

[15] Web Services Architecture,W3C Working Group Note 11 February 2004, http://www.w3.org/TR/ws-arch/.

[16] Ian H. Witten and Eibe Frank, *Data Mining: Practical machine learning tools with Java implementations*, Morgan Kaufmann, San Francisco, 2000.

Appendix:

Table 1. COI Service Layer

Service name	Capabilities and functionality	Technologies used
Membership Management Services (MMS)	• Provide mechanisms for users to register/un-register with specific COIs, manages connection among individual COIs	Directory service, Database indexing, Searching
Policy Specification/Evaluation Services (PSS)	• COI framework supports policy-based computing. Policies are specified in PSEL and policies can be extended in real time at runtime to determine system properties such as security and reliability	Policy Specification, Policy execution, Policy enforcement
Situation Awareness Services (SAS)	• Collect data from sensors and other services, analyze the data received, issue alerts to concerned COIs and participants	Data mining, Data analysis, Classification, Pattern detection
Dynamic Reconfiguration Services (DRS)	• Support dynamic reconfiguration in SOA in real time at runtime • The COI has DRS that can continuously monitor each participating service, initiate reconfiguration process, and implement using the C2 policies • Services can be added, removed at runtime without interrupting the system operations	Directory service, Distributed monitoring, Data mining, Profiling, Authentication
Data Classification Services (DCS)	• Classify data collected using data mining approach to support decision making	Data mining Decision making
Service Mining Services (SMS)	• Dynamically and optimally match a request to existing services	Data mining Operations Research

Table 2. Support Service Layer

Service name	Capabilities and functionality	Technologies used
Discovery Services (DIS)	• Provides discovery in traditional SOA such as .NET • In addition to static information, these discovery service also needs dynamic information about available services in the network including their operational profile, current status, scheduled tasks, maintenance schedule and backup data information	Naming and directory services
Messaging Services (MES)	• Provide event-driven messaging to COIs triggered by services such as policy, discovery, situation awareness & dynamic reconfiguration, security and storage which form complex hierarchy with attributes of reliability, survivability, security & non-repudiation, timely delivery	Communication
Security Services (SES)	• Services ensures system security including encryption, authentication, filtering and security policy enforcement	Encryption, Security models
Scenario Specification Services (SSS)	• Allow users to specify requirements in a scenario specification language and ACDATE model which helps in system verification, simulation, C2 policy enforcement, completeness and consistency checking	Scenario model, Scenario specification, Scenario execution, Simulation, Completeness and Consistency
Test and Evaluation Services (TES)	• Supports automated test case/scripts generation using pattern-driven techniques and state model • Support distributes test execution	Test script generation, Verification, Interface specification
Simulation Services (SIS)	• Once system scenarios are specified using ACDATE model, these services simulate system behavior to allow various runtime analyses such as security analysis, concurrency analysis, timing and completeness and concurrency analysis	Distributed simulation, code generation, concurrency
Data Acquisition Services (DAS)	• Collect data for situation awareness analysis, runtime monitoring and evaluation, security analysis and test and evaluation	Semantic web mining, Data warehousing
Storage Services (STS)	• Provide data storage and retrieval capabilities	Databases, Data clustering, Data storage, Information retrieval